Asc lab 4 dezastru

Recapitulare lab 3

→array-uri de elemente de tip .long

→apelul procedurii printf

movl $v, %edi

daca %ecx retine un index din array atunci accesarea elementului este

movl (%edi, %ecx, 4), %ebx

echiv cu v(, %ecx,4), %ebx

→printf (formatString, arg…) (%d-intregi, %f- floats, %s- siruri)

Printf(“S-au citit %d si %d \n”)

Push y

Push x

Push $formatStr

call printf

pop %ebx

pop %ebx

pop %ebx

Apelul procedurii scanf

Scanf(formatScanf, arg…..)

Format Scanf → la fel cu cel de la printf

Citirea intregilor

.data

X: .space 4

formatScanf: .asciz “%d”

Scanf(“%d”,&x) //in c

.text

Push $x

Push $formatScanf

Call scanf

Pop %ebx

Pop %ebx

Movl x, %eax

→Siruri de caractere

Array-uri de elemente de tip .byte

.data

Str: .asciz “Sir de caractere”

.text

.global main

Main:

Movl $str, %edi

Movl $0, %ecx

Et\_loop:

Movb (%edi, %ecx, 1), %ah

Cmp $0, %ah

Je et\_exit

//prelucrare ah

Inc %ecx

Jmp et\_loop

Eax (32b, 4B) – AX- 16b 2B

-AL 8b 2B

-AH 8b 1B

AL, AH, BL, BH, CL ,CH, DL, DH

%ah

Cmp $65, %ah

Cmp $’A’, %ah

Je sau jne

Se citeste de la stdin (prin apel scanf) un sir de caractere fara spatii de maxim 50 de caractere. Sa se determine numarul de vocale si sa se afiseze acest nr la STDOUT printr-un apel la printf

.data

formatStr: .asciz "Vocale: %d\n"

scanfStr: .asciz "%s"

x:. space 49

n:. long 0

.text

.global main

main:

push $x

push $scanfStr

call scanf

pop %ebx

pop %ebx

movl $str, %edi

movl $0, %ecx

et\_loop:

movb(%edi, %ecx, 1), %ah

cmp $0, %ah

je et\_exit

cmp $'a',%ah

je et\_at

cmp $'e',%ah

je et\_at

cmp $'i',%ah

je et\_at

cmp $'o',%ah

je et\_at

cmp $'u', %ah

je et\_at

incl %ecx

jmp et\_loop

et\_at:

add $1, n

incl %ecx

jmp et\_loop

et\_exit:

push $n

push $formatstr

call printf

pop %ebx

pop %ebx

ASC LAB 5

Proceduri in limbaje de asamblare pt x86

→cadrul de apel este construit prin intermediul stivei

Stiva: push x

Pop x

Stiva este gestionata de %esp -stack pointer

Pe stiva putem pune doar elemente pe 4B (%esp se modifica din 4 in 4)

→ Stiva creste spre adrese mai mici

→Stiva scade spre adrese mai mari

0(%esp) este intotdeauna elemental din varful stivei

Urmatorul element pe stiva(in jos) este 4(%esp)

→Push x:

Sub $4, %esp

Mov x, 0(%esp)

→Pop: (nu e definit in limbajele de asamblare)

Add $4, %esp

→Pop x:

Mov 0(%esp), x

Add $4, %esp

Conventii de implementare a procedurilor

1. Argumentele procedurilor trebuie date prin stiva, de la dreapta la stanga
2. Valorile returnate de procedura se vor pune in ordine in registry %eax, %ecx, %edx, iar apoi pe stiva

%eax=min(x,y)

Printf

Apelul procedurilor

→Instructiunea call

Call <label> (Diferenta dintre call si jump este ca dupa apelul call revine controlul asupra programului exact din punctul in care eram)

Main:

Push y

Push x

Call suma

Pop %ebx

Pop %ebx

Suma:

…